ANOTHER ASYMPTOTIC NOTATION : "ALMOST"

NABARUN MONDAL AND PARTHA P. GHOSH

Abstract. Asymptotic notations are heavily used while analysing runtimes of algorithms. Present paper argues that some of these usages are non trivial, therefore incurring errors in communication of ideas. After careful reconsideration of the various existing notations a new notation is proposed. This notation has similarities with the other heavily used notations like Big-Oh, Big Theta, while being more accurate when describing the order relationship. It has been argued that this notation is more suitable for describing algorithm runtime than Big-Oh.

1. The Problem At Hand

Describing the exact runtime of an algorithm [3][5][6] is not a trivial task. Hence, the effort is directed upon finding a function that approximates the runtime. To describe the approximate runtime for the algorithms, the concept of asymptotic notations were borrowed from Number Theory [1][2][4]. In this section we discuss the most common of the asymptotic notations, introduced by Bachmann-Landau [1][2], and describe the semantic problem with the overly used Big-Oh notation[3][5][6][4].

Definition 1.1. ***Big-Oh : O(.)***
Let $f(n)$ and $g(n)$ be functions such that :-

$$\exists k > 0 \; \exists n_0 \; \forall n > n_0 \; |f(n)| \leq |g(n) \cdot k|$$

then $f(n) \in O(g(n))$ or with some abuse of notation $f(n) = O(g(n))$.

Informally this stands for f is bounded above by g (up to constant factor) asymptotically.

Definition 1.2. ***Small-Oh : o(.)***

2010 *Mathematics Subject Classification.* Primary 68W40 ; Secondary 03D15 , 68Q15 .
Key words and phrases. Algorithms ; Runtime ; Asymptotic Notations ; Big Oh ; Big Theta ; Almost .
Nabarun Mondal : Dedicated to my late professor Dr. Prashanta Kumar Nandi.
Dedicated to my parents.
Big thanks to:- Abhishek Chanda : You always have been constant support.
In Memory of : Dhrubajyoti Ghosh. Dear Dhru, rest in peace.
Partha. P. Ghosh : Dedicated to my parents and family without their presence we are nothing.

Let $f(n)$ and $g(n)$ be functions such that :-

$$\forall k > 0 \; \exists n_0 \; \forall n > n_0 \; |f(n)| \leq k \cdot |g(n)|$$

then $f(n) \in o(g(n))$ or with some abuse of notation $f(n) = o(g(n))$.

Informally this stands for f is dominated by g (up to constant factor) asymptotically.

Theorem 1.1. ***Relation between Big-O and Small-o.***
If $f \in o(g)$ then $f \in O(g)$.

Proof. We compare the definitions (1.2) and (1.1). Note that the $f \in o(g) \implies \exists k$ for which $|f(n)| \leq |g(n) \cdot k|$. Hence the given is proved. □

This theorem (1.1) creates a confusion about the way mathematicians use the big-oh notation $f = O(g)$ and computer engineers use them. Let $f \in O(g)$, then from Computer Engineering standpoint :-

"growth rate of f is of the order of g"

and it's precise mathematical meaning:-

"growth rate of f is less than or equal to the order of g"

We now demonstrate the confusion with a real example. For any algorithm $\mathcal{A}$ let the average runtime complexity be a function '$A(n)$' and worst case runtime complexity be another function '$W(n)$' where 'n' be the input size. In the *strictly mathematical* way $A \in O(W)$. However, this becomes counter intuitive as the intuitive notion of big-O as *the order of the expression* as it is majorly used in Engineering.

Example of Quick sort can be used to demonstrate the point. $A(n) \approx nLog_2 n$ and $W(n) \approx n^2$. *Strictly mathematically* :-

$$nLog_2(n) = O(n^2) = O(n^3) = O(n^k) \; ; \; k \geq 2$$

because

$$nLog_2(n) = o(n^2) = o(n^3) = o(n^k) \; ; \; k \geq 2$$

and $f = o(g) \Rightarrow f = O(g)$. However, clearly the order-of $nLog_2(n)$ is not the order of n^2 or n^3 etc, and the confusion arises because technically $O(g)$ family is not an open family of functions (it depicts the relation '$\leq$' in some sense). The family of functions $o(g)$ is however an open family, depicting the relation '$<$' [4][3]. Simply put Big-Oh is not the *of the order of* function family.

2. Stricter Asymptotic Bounds

This problem of confusion on notation with intuition can be avoided by using any stricter notion, which should match with the intuition of *growth order*. One such Bachmann-Landau notation is the Big-Theta notation which is described next.

Definition 2.1. ***Big-Theta : $\Theta(.)$***
Let $f(n)$ and $g(n)$ be functions such that :-

$$\exists k_1 > 0 \; \exists k_2 > 0 \; \exists n_0 \; \forall n > n_0$$

$$g(n) \cdot k_1 \leq f(n) \leq g(n) \cdot k_2$$

then $f(n) \in \Theta(g(n))$ or with abuse of notation $f(n) = \Theta(g(n))$.

This stands for f is bounded both above and below by g asymptotically. For example given that the function $f(n) = n^3 + 1000n^2 + n + 300$ then, $f \in o(n^5)$ as well as $f \in O(n^5)$. But $f \in \Theta(n^3)$ so is $f \in \Theta(n^3 + n^2)$, so on and so forth. Therefore, the notation $\Theta(.)$ allows one to be far more accurate than that of the more popular $O(.)$ notation, but still, not enough accurate. One simple example to show this would be $f(n) = 2 - sin(n)$ while $g(n) = c$ a constant function with $c > 0$. Clearly $f = \Theta(g)$ but, then, accuracy is lost here.

One way to resolve this loss of accuracy is to find an equivalence or similar notation, the real *on the order of* notation as defined next.

Definition 2.2. ***On the Order Of : $\sim$***
Let $f(n)$ and $g(n)$ be functions such that :-

$$\forall \varepsilon > 0 \; \exists n_0 \; \forall n > n_0 \; \left| \frac{f(n)}{g(n)} - 1 \right| < \varepsilon$$

then, $f \sim g$.

The problem with definition 2.2 is that this is much of a strict bound. It might not be optimal to find a function g that approximates f till the exact same growth order. For example take the function $f(n) = 3n^3 + 2n^2 + n$. We have $g(n) = 3n^3$ which makes $f \sim g$ but for another $g'(n) = n^3$ $f \not\sim g'$. Constant factors should be ignored, in this case it is not. We note that, however, another way of representing the same notation is:-

$$f \sim g \implies \forall n > n_0 \; \frac{f(n)}{g(n)} \to 1$$

Replacing the constant 1 with a varying parameter K makes it into a better asymptotic notation independent of the constant factor. But that factor, should never be 0 or ∞.

Therefore, we can define a new, but similar asymptotic notation *"Almost"* .

Definition 2.3. ***Almost : a(K,.).***
A function $f(n)$ is almost $g(n)$ or $f \in a(K, g)$ iff:-

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = K \; ; \; 0 < \; K < \infty$$

exists.

3. Properties of "Almost"

In this section we establish properties of "almost" with theorems which relates the notation of *almost* (definition 2.3) with other notations.

The trivial properties first:-

(1) $f \in a(1, f)$, that is $f \; a \; f$.
(2) $f \in a(K, g) \implies g \in a(1/K, f)$ which is $f \; a \; g \implies g \; f \; a$.
(3) $f \in a(K_1, g)$ and $g \in a(K_2, h)$ then $f \in a(K_3, h)$ with $K_3 = K_1 K_2$ that is $f \; a \; g$ and $g \; a \; h \implies f \; a \; h$.
(4) $f \in a(1, g) \implies f \sim g$.

Theorem 3.1. $a(.,.)$ ***is Equivalence Class Relation.***

Functions related using the notation $a(.,.)$ (definition 2.3) are an equivalence class of functions.

Proof. The first three trivial properties demonstrates this. □

Theorem 3.2. ***Relation of*** $a(.)$ ***with*** $O(.)$***.***

If $f \in a(K, g)$ then $f \in O(g)$ but not vice versa.

Proof. If we have

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = K$$

We note that $f \in a(k, g)$ implies $0 < K$, while $f \in O(g)$ implies $\exists k \; ; \; 0 \leq k$. Which establishes the theorem. □

Theorem 3.3. ***Relation of*** $a(.)$ ***with*** $\Theta(.)$***.***

(1) If $f \in a(K, g)$ then $f \in \Theta(g)$.
(2) If $f \in \Theta(g)$ then it is not necessary to have $f \in a(K, g)$.
(3) Therefore, relation $a(.)$ is a subset of $\Theta(.)$.

Proof. By definition we have for $n > n_0 \; f(n) = Kg(n)$. That is, obviously $f < (K+1)g$ and clearly $f > (K-1)g$. Which means, tallying with the definition 2.1, we have $k_1 = K - 1$ and $k_2 = K + 1$. Which proves the first part.

We show the second part by bringing in one example. Let $f(n) = 2 - sin(n)$. Clearly then $\sup(f(n)) = 3$ and $\inf(f(n)) = 1$. Obviously then we can define $g(n) = 1$, with $k_1 = 1$ with $k_2 = 3$, so that :-

$$k_1 g \leq f \leq k_2 g$$

and therefore $f \in \Theta(1)$. However, no limit exists for the ratio $f(n)/g(n)$ with $n \to \infty$, and therefore $f \notin a(k, g)$.

These show that there are functions $f = \Theta(g)$ but $f \neq a(, g)$, hence, the relation between the family is depicted as $a \subset \Theta$. □

4. Summary : Advantage of using "almost"

There is no way one can state $f \in a(K, g)$, unless, they are comparable to the limit, as defined by definition 2.3.

Therefore, the biggest advantage of the proposed notation is less confusion in the usage of the notation, because *almost* is an equivalence notation. Take for example $n^2 \in O(n^3)$ but $n^3 \notin O(n^2)$. But clearly $n^2 \notin a(K, n^3)$.

In the same note, one can not in general write $A = a(K, W)$ like in section 1 , unless, of course A, W are really comparable, as in merge-sort. In specificity, we can not use the notation $a(K, .)$ to compare $f = nLog_2(n)$ and $g = n^2$, in case of Quick Sort.

Given the limit at infinity exists, *almost* becomes both the tight upper and lower bound, that is $\Theta(.)$ from definition 2.1 using the theorem 3.3. On the other hand, if no such limit exists (example given is theorem 3.3), we can still use the original $\Theta(.)$ notation.

In summary, the notation $a(K, .)$ has advantages borrowed from all the notations, without any of their shortcomings, as long as it can be defined. Also, this is not as much a tight bound compare to the notation $\sim$ (definition 2.2).

References

[1] Paul Bachmann, *Die Analytische Zahlentheorie. Zahlentheorie. pt. 2* . 1894.
[2] Edmund Landau, *Handbuch der Lehre von der Verteilung der Primzahlen. 2 vols*. 1909.
[3] Donald Knuth, *The Art of Computer Programming, Volume 1: Fundamental Algorithms, Third Edition*. Addison-Wesley, 1997. ISBN 0-201-89683-4. Section 1.2.11: Asymptotic Representations, 107-123.
[4] Graham, Ronald L. ; Knuth,Donald E. ; Patashnik,Oren. , *Concrete Mathematics: A Foundation for Computer Science* . Addison-Wesley Professional; 2 edition (March 10, 1994).
[5] Cormen, Thomas H. ; Leiserson,Charles E. ; Rivest,Ronald L. ; Stein, Clifford., *Introduction to Algorithms* . The MIT Press; third edition edition (July 31, 2009).
[6] Aho; Hopcroft ; Ullman , *Data Structures and Algorithms* . Addison-Wesley; 1st edition (January 11, 1983).

D.E.Shaw & Co. India, Hyderabad
E-mail address: mondal@deshaw.com

Microsoft India, Hyderabad
E-mail address: parthag@microsoft.com